BAB IV

PENERAPAN PTES

Pada BAB IV akan diterapkan 6 langkah PTES pada iRaise saat ini, yaitu Pre-engagement, Intelligence gathering, Threat modeling, Vulnerability analysis, Exploitation, Post exploitation.

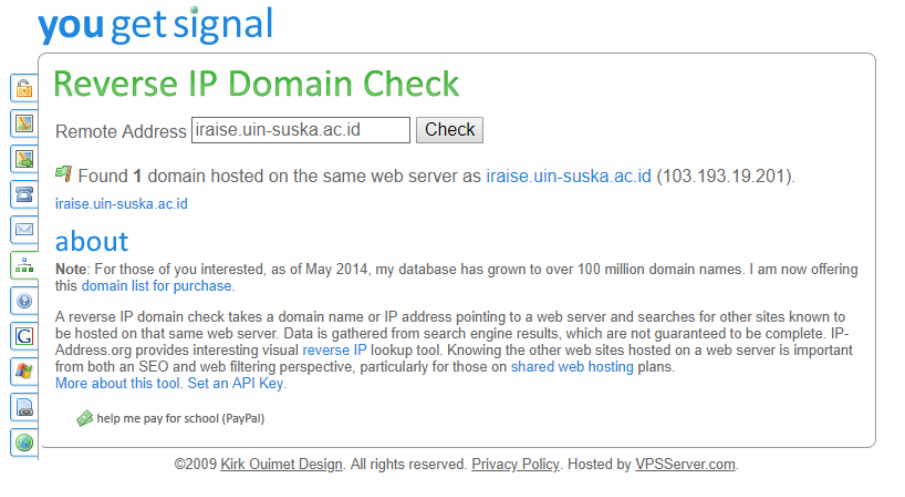
4.1. Pre-engagement

Pada tahapan ini dilakukan pembuatan surat izin untuk melakukan *penetration testing* terhadap iRaise ke pihak PTIPD yang berkaitan dengan batasan yang pengujian metode serangan *penetration testing*. Persetujuan dari pimpinan PTIPD terhadap penetration testing pada iRaise tertera pada lampiran.

4.2. Intelligence gathering

Pada tahapan ini yaitu mengumpulkan informasi penting terhadap iRaise, informasi yang didapatkan bisa menjadi acuan untuk melakukan penetration testing. Informasi yang didapatkan seperti IP address, reverse ip, dns, dan informasi lainnya yang diharapkan bisa membantu penetration testing. Pada tahap ini, dengan menggunakan *nmap* penguji mendapati IP dari webstite iRaise yang beralamatkan iraise.uin-suska.ac.id yaitu 103.193.19.201. IP tersebut dapat diguakan untuk mencari info selanjutanya yaitu *reverse* IP dari iRaise.

Selanjutnya *reverse* ip yaitu pencarian apakah dalam sebuah server terdapat lebih dari 1 *website*, apabila didalam server iraise terdapat *subdomain* lain iraise akan sangat rentan terhadap serangan *jumping server*. *Jumping server* adalah tindakan penyerang untuk melakukan lompatan kedirektori lain yang berada dalam 1 server. Dalam pencarian *reverse ip* menggunakan bantuan *website* dengan *url* https://www.yougetsignal.com/tools/web-sites-on-web-server. Hasil yang didapat dari *website* tersebut dapat dilihat pada gambar 4.2 menyatakan iraise hanya berdiri sendiri dan berarti tidak dapat dilakukan *jumping server*



Gambar 4.1 Pengecekan Reverse IP

Selain itu berdasarkan hasil dari *whois* dapat dilihat pada gambar 4.3 diketaui juga bahwa domain uin-suska.ac.id dibeli dan menggunakan dns dari Rumah Web yang merupakan salah satu penyedia layanan hosting dan domain. Hal ini juga sangat rentan terhadap *social engginering*.

Gambar 4.2 Pengecekan Whois

4.3. Threat modeling

Pada tahap ini mencari pemodelan ancaman yang akan digunakan untuk melakukan *penetration testing* terhadap iRaise. Pada tahapan ini penyerang menggunakan 10 model ancaman yang telah tersedia pada OWASP TOP 10 2017 yang merupakan 10 model ancaman yang paling banyak terjadi pada sebuah *website* pada umumnya dan juga sesuai dengan perjanjian pada tahap *Pre-Engagement*, yaitu diantaranya :

1. SQL *Injection*
2. *Broken Authentication*
3. *Sensitive Data Exposure*
4. XXE
5. *Broken Acces Control*
6. *Security Miss Configuration*
7. XSS
8. *Insecure Deserialization*
9. *Using Component* with *Vulnerabilites*
10. *Insufficien Logging* & *Monitoring*

Penguji menggunakan OWASP Top 10 dikarenakan ini adalah proyek yang meliputi pakar keamanan dari seluruh dunia untuk menghasilkan 10 daftar ancaman paling kritis terhadap keamanan aplikasi web.

4.4. Vulnerability analysis

Tahap ini merupakan tahapan untuk mendapatkan informasi lebih spesifik sebelum memasuki tahapan *explotation*. Pada tahapan ini penguji akan menggunakan beberapa *tools* yang digunakan untuk melakukan vulnerability analysis. Penguji menggunkana 2 *tools* yang tersedia pada Kali Linux.

Kali Linux adalah sebuah sistem operasi yang dikhususkan untuk melakukan *penetration testing.* Tools yang akan digunakan adalah Nikto dan

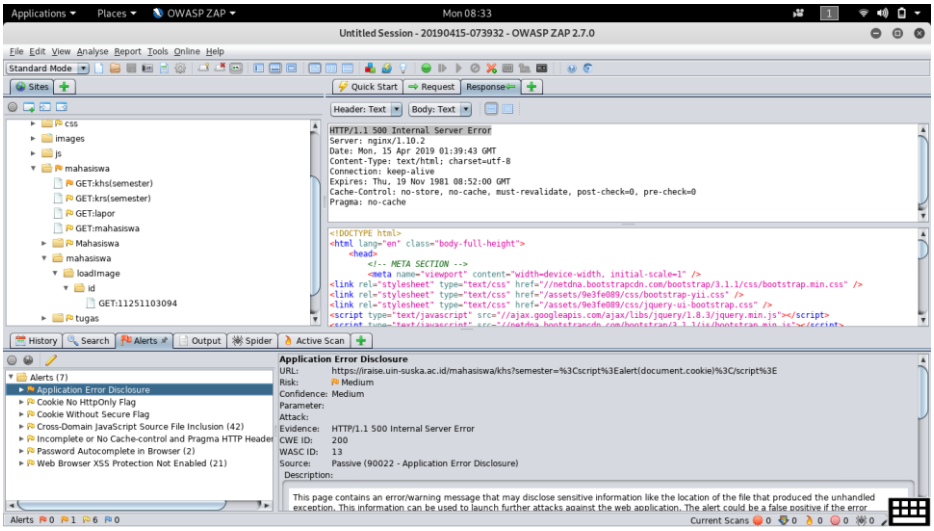
OWASP Zap. Kedua tools tersebut merupakan tools Vulnerabilty Analysis yagn tersedia langsung pada sistem operasi Kali Linux.

Sebelum menerapkan 10 ancaman OWASP, dilakukan scan iRaise menggukan Nikto dan ditemukan celah XSS pada hasil scan menggunakan Nikto seperti pada gambar 4.3

Gambar 4.3 *Scanning* Menggunakan Nikto

Pada gambar 4.3 dengan perintah “nikto -h iraise.uin-suska.ac.id -ssl -Tuning 9” perintah tersebut berfungsi untuk memerintahkan nikto untuk melakukan scan terhadap host iraise.uin-suska.ac.id menggunakan SSL karena iRaise telah menggunakan SSL, dan untuk *Tuning* 9 merupakan fungsi untuk melakukan pengecekan yang lebih menyeluruh terutama celah SQL Injection. Pada hasil scan nikto iRaise memiliki masalah keamanan pada celah XSS dan memberi tahukan IP local server iRaise. Untuk SQL Injection tidak tertera pada hasil nikto walaupun sudah menggunakan Tuning khusus SQL Injection.

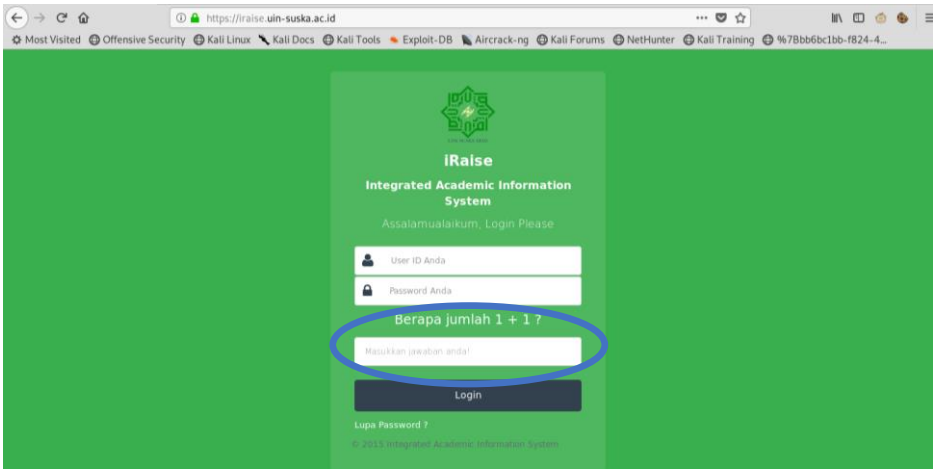
Setelah menggunakan Nikto penguji juga mencoba melakukan scan menggunakan OWASP Zap. Penguji juga mendapati hal sama yaitu celah XSS yang dapat dilihat pada gambar 4.4



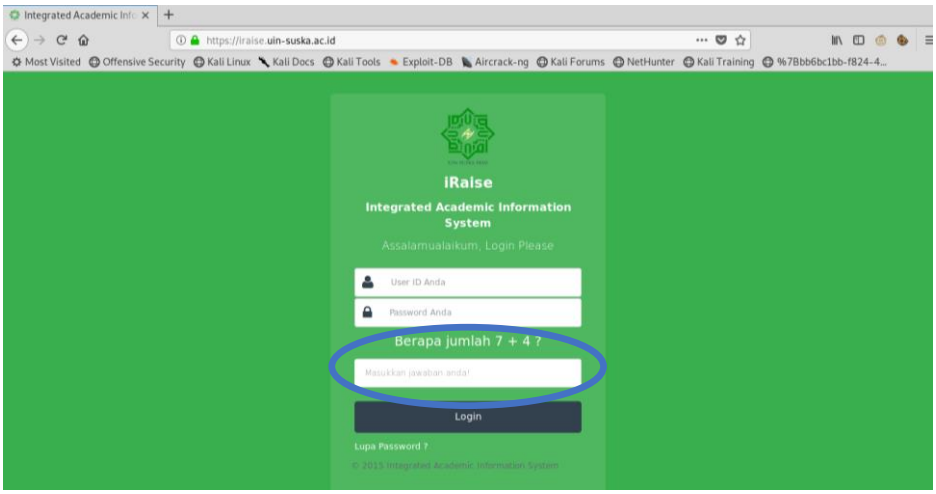
Gambar 4.4 Hasil OWASP Zap

Dari hasil scan menggunakan 2 aplikasi tersebut diketahui iRaise mempunyai celah XSS, XSS berada pada nomor 7 jenis ancaman paling kritis yang terdapat pada OWASP TOP 10 Tahun 2017. Dan selanjutnya akan dilakukan analisa terhadap 10 ancaman paling berbahaya dari OWASP Top 10 Tahun 2017

1. SQL *Injection*, iRaise bisa dikategorikan aman dari serangan ini. Hal tersebut juga telah dibenarkan oleh pihak YII bahwa celah SQL Injection memang pernah bisa dilakukan tapi celah tersebut telah dilakukan *patching* pada versi yang digunakan di iRaise. YII yang bisa dilakukan serangan SQL Injection adalah YII versi 1, sedangkan iRaise telah menggunakan YII versi 2. Bukan hanya dari YII tapi penguji telah melakukan percobaan dan hasil dari *scanning* Nikto pada tahapan sebelumnya yang berfokus kepada SQL Injection tidak disebutkan pada hasil Nikto.
2. *Broken Authentication*, iRaise juga bisa dikategorikan aman dari jenis serangan ini, berdasarkan modul dari OWASP, jenis aplikasi yang bisa diserang dengan serangan ini adalah aplikasi yang bisa di *Brute Force* sedangkan iRaise tidak bisa dilakukan serangan tersebut dikarenakan iRaise memiliki verifikas yaitu persoalan matematika random yang berganti – ganti untuk melakukan login.

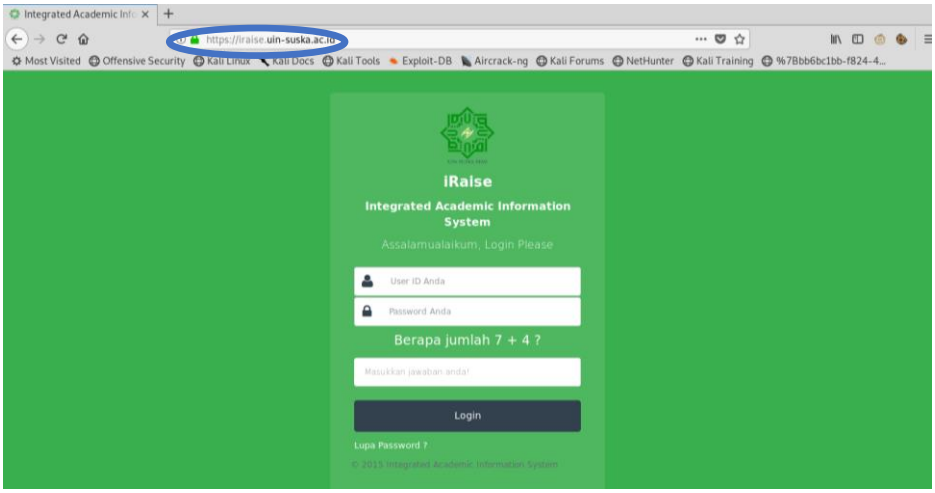


Gambar 4.5 Persoalan Matematika iRaise 1



Gambar 4.6 Persoalan Matematika iRaise 2

3. *Sensitive Data Exposure*, ini merupakan celah keamanan yang sering dimanfaatkan melalui serangan Man-in-the-middle Attack (MITM). Untuk menghentikan serangan MITM sebuah *website* harus menggunakan SSL/HTTPS dan iRaise telah menggunakan SSL/HTTPS yang bisa dilihat pada gambar 4.7, oleh karena itu iRaise bisa dikatakan tidak dapat diserang menggunakan ancaman *Sensitive Data Exposure*.



Gambar 4.7 SSL/HTTPS iRaise

4. XML *External Entities*, biasa disebut dengan serangan XEE. Serangan ini hanya berlaku untuk aplikasi web yang berbasis XML, sedangkan iRaise bukanlah web aplikasi berbasis XML. Oleh karena itu jenis serangan ini tidak bisa dilakukan di iRaise.
5. *Broken Access Control*, ancaman ini dapat diexploitasi dengan melakukan percobaan secara manual tidak menggunakan *tools* dan akan dilanjutkan pada tahap *exploitation* untuk mengetahui iRasie memilike ancaman ini atau tidak.
6. *Security Misconfiguration* ini merupakan ancaman yang berhubungan dengan jenis ancaman nomor 5, apabila *developer* membiarkan konfigurasi default pada sebuah keamanan *website*, dengan sedikit percobaan secara manual akun tersebut dapat melakukan akses yang tidak seharusnya bisa diakses.
7. XSS, yaitu *Cross-Site Scripting* dan berdasarkan hasil scan nikto sebelumnya iRaise dapat diserang menggunakan serangan XSS. Dan akan dilanjutkan di tahap *exploitation.*
8. *Insecure Deserialization* yaitu ancaman yang paling sering terjadi pada web aplikasi berbasis XML dan JSON. Ancaman ini hamper serupa dengan ancaman nomor 4 yaitu ancaman terhadap *website* berbasi XML. Oleh karena itu bisa dikatakan iRaise bebas dari ancaman nomor 8 dikarenakan iRaise bukanlah website berbasis XML.
9. *Using Components with Known Vulnerabilities*, yaitu melakukan serangan dengan celah yang telah diketahui sebelumnya, berhubung sebelumnya iRaise

bisa dilakukan *Session Hijacking* pada saat kerja praktek, ancaman *Sesion Hijack* selanjutnya akan dilanjutkan kembali pada saat *exploitation*.

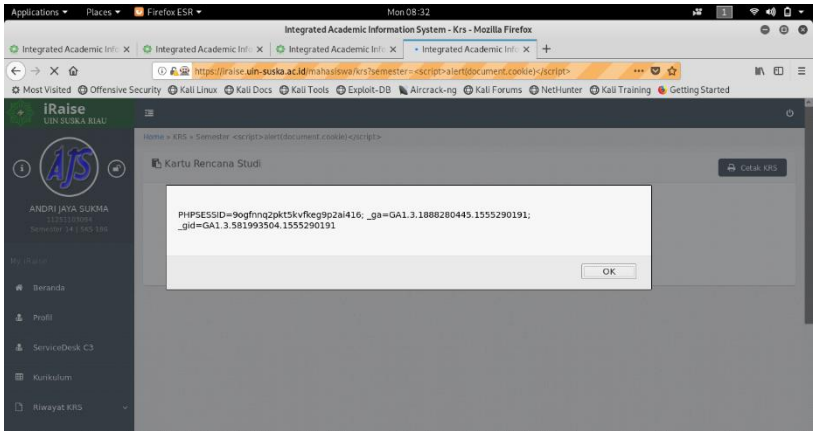
10. *Insufficient Logging and Monitoring* merupakan tidak adanya pengawasan atau bahkan respon dari sebuah aplikasi apabila ada percobaan login yang salah berkali – kali. Ini membuat seorang *penetration tester* bebas mealakukan percobaan serangan dan apabila tidak ada pengawasan atau notifikasi login atau notifikasi adanya tindakan tidak semesetinya, dapat membuat user aslinya atau *admin* iRaise tidak mengetahui bahwa ada yang berusaha atau mengontrol akun yang dia miliki. Untuk ancaman ini hanya dapat dilihat setelah pelakukan *penetration testing* apakah iRaise memiliki log aktifitas *percobaan penetration* testing atau tidak.

4.5. Exploitation

Tahap ini merupakan tahapan dimana penguji melakukan percobaan *penetration testing* berdasarkan info yang telah didapatkan pada tahapan sebelumnya, berdasarkan hasil dari tahapan sebelumnya iRaise memiliki celah XSS, dan juga akan melakukan ancaman *broken access control*, *security misconfiguration,* dan *using component with know vulnerabilities*.

4.5.1. Percobaan XSS

Pada saat melakukan *exploitation* setelah mencoba beberapa halaman, akhirnya ditemukan celah XSS pada halaman KRS mahasiswa dengan memasukkan kode "*<script>alert(document.cookie)</script>"* pada URL, yang dapat dilihat pada gambar 4.8



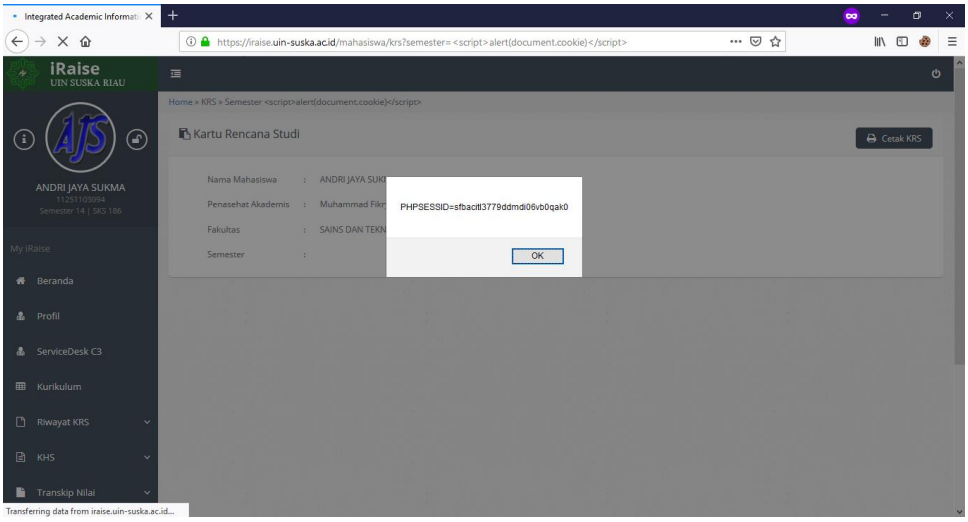
Gambar 4.8 Percobaan XSS

XSS pada gambar diatas membuktikan iRaise masih memiliki celah XSS terbukti benar. Dari kode yang masukkan, berisi perintah memperlihatkan *cookie* seorang pengguna.

4.5.2. **Percobaan Session Hijacking**

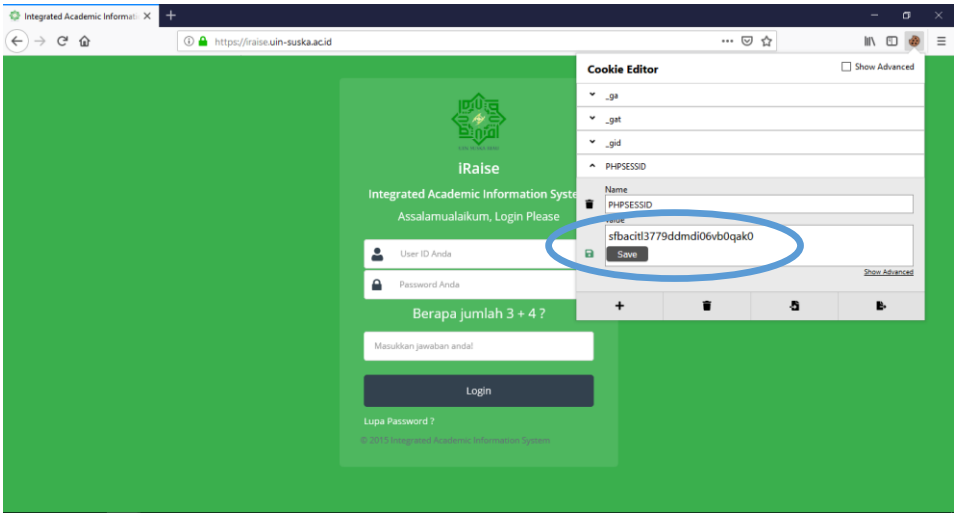
Selanjutnya berdasarkan hasil tahapan sebelumnya pada ancaman nomor 9 diketahui sebelumnya iRaise memiliki celah keamanan *Sesion Hijacking* yaitu celah dimana peretas memanfaatkan sebuah sesi yang dapat membuatnya memasuki halaman yang bisa diakses ketika sudah login tanpa harus mengetahui username atau password orang tersebut.

Dengan bantuan *plugin Cookie Editor* yang terdapat pada *browser*, *cookie* yang didapatkan dari XSS sebelunya bisa dimanfaatkan untuk mendapatkan akses tanpa harus *login*. Disini penguji kembali mencoba mendapatkan *cookie* dari celah XSS pada gambar 4.9



Gambar 4.9 Serangan XSS

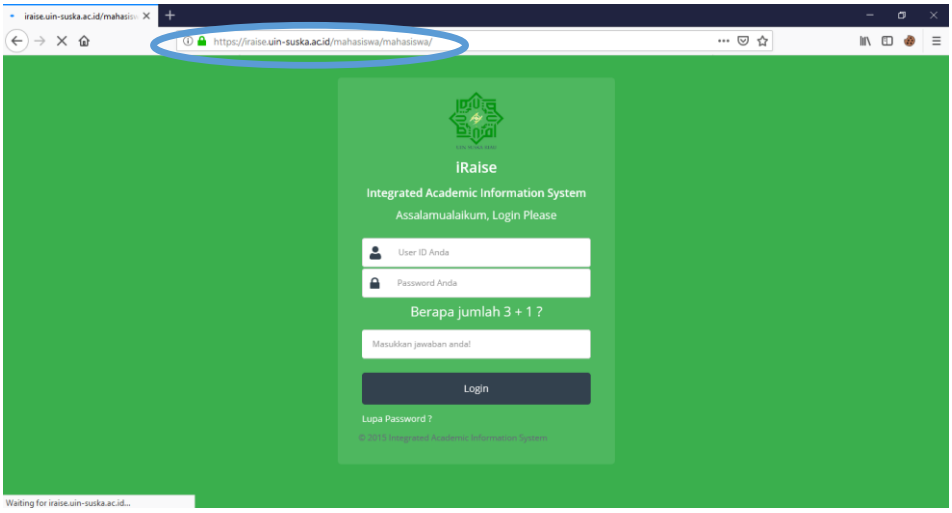
Dari hasil gambar 4.9 didapatkan sebuah *cookie* PHPSESSID yang memiliki nilai "sfbacitl3779ddmdi06vb0qak0". Selanjutnya buka browser baru yang tidak memiliki sesi login iRaise, disini penguji menggunakan mode penyamaran untuk mendapatkan sebuah sesi dan kembali pada mode biasa untuk memasukkan sesi yang telah didapatkan dengan bantuan *Cookie Editor* bisa dilihat pada gambar 4.10



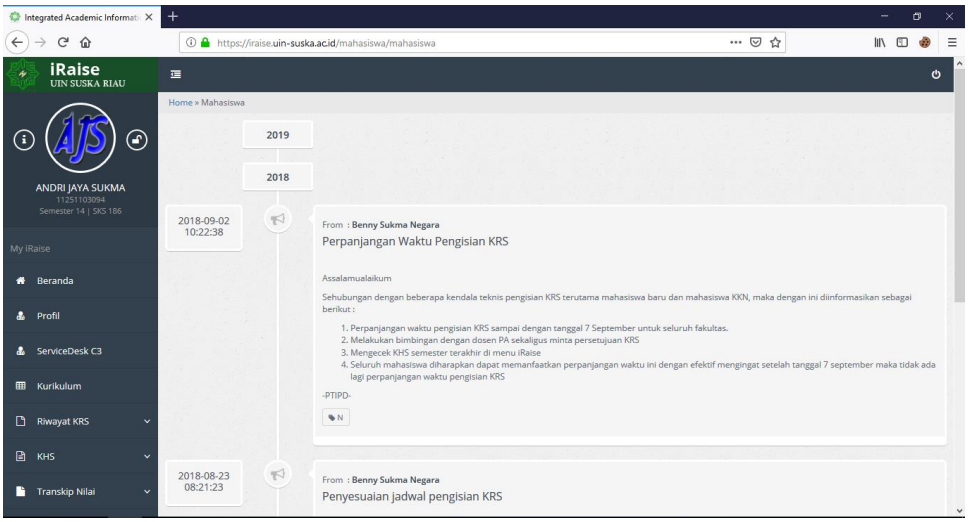
Gambar 4.10 Simpan Perubahan Cookie

Setelah memasukkan isi *cookie* dan disimpan, langkah selanjutya hanya dengan menekan tombol F5 untuk melakukan *reload* halaman *login* tersebut akan langsung membawa pada halaman utama mahasiswa karena tadi mendapatkan sesi mahasiswa bisa dilihat pada gambar 4.11 dan gambar 4.12 proses *reload* masih

berlangsung tapi halaman *url* sudah berganti dan dialihkan ke halaman utama mahasiswa tanpa memasukkan *username* dan *password.*



Gambar 4.11 Proses Loading Halaman Tanpa Login



Gambar 4.12 Login Sukses Tanpa Username dan Password

Setelah dilakukan percobaan *session hijacking* dan ternyata masih bisa dimanfaatkan. Untuk melanjutkan langkah selanjutnya penguji mencoba melakukan *sniffing* untuk mendapatkan c*ookie* seorang admin atau siapa saja yang memiliki tingkatan diatas seorang mahasiswa.

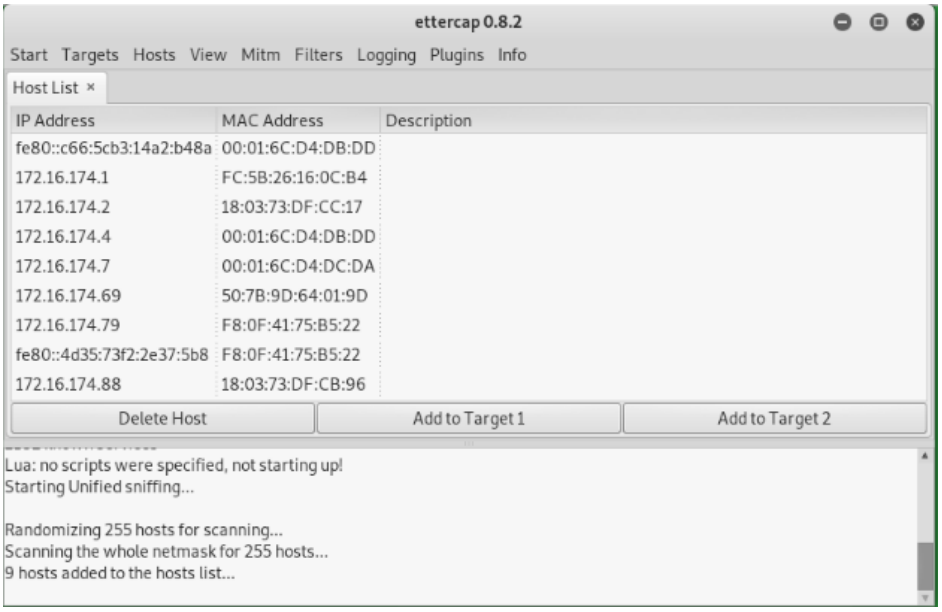
4.5.3. **Percobaan *Sniffing***

Percobaan *sniffing* dilakukan dari dalam PTIPD, karena admin atau seorang yang memiliki akses yang tinggi tentu saja berada di PTIPD. Percobaan sniffing ini

memanfaatkan seorang admin dari iRaise login. Pada tahapan ini penguji melakukan *sniffing* di C3 PTIPD karena banyak mahasiswa yang ingin mengganti password mereka. Dari sini tentu saja seorang admin akan melakukan login iRaise dengan akun yang memiliki akses lebih untuk mengganti password seorang mahasiswa.

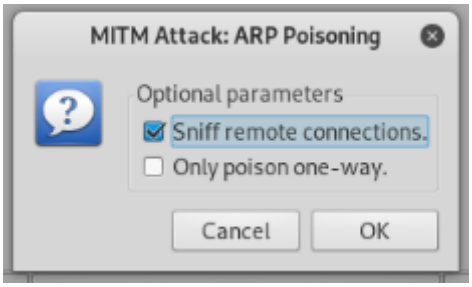
Percobaan sniffing dilakukan menggunaka beberapa *tools*, seperti *wireshark, ettercap,* dan sedikit perintah dari terminal linux. Percobaan *sniffing* akan dilakukan dari dalam iRaise dan berada pada satu jaringan dengan salah seorang admin.

1. Melakukan *scanning* menggunakan *ettercap* dengan memasukkan semua hasil *scan* menjadi target untuk *sniffing* seperti gambar 4.13



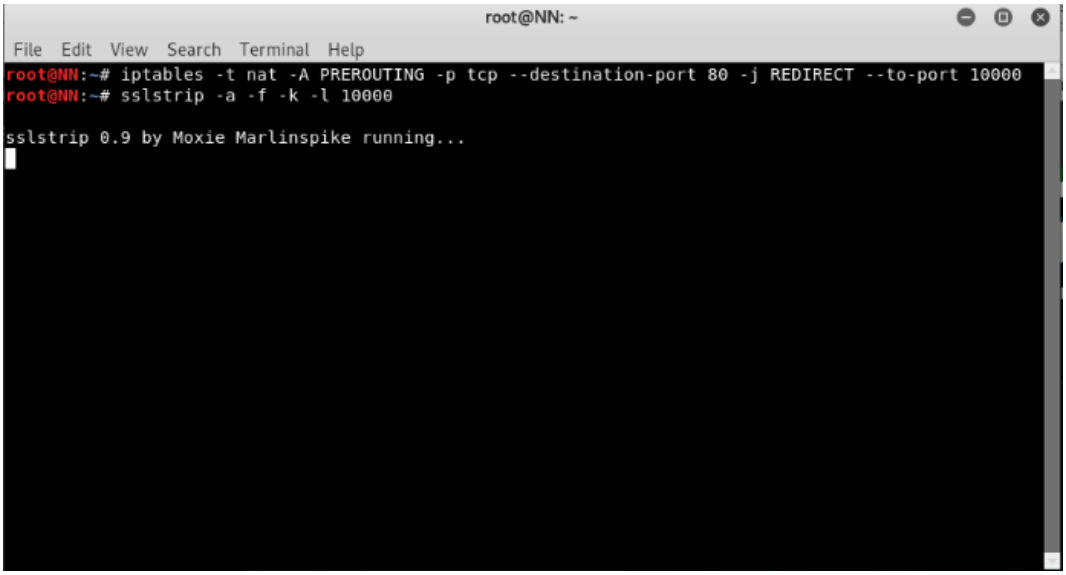
Gambar 4.13 Menambahkan *Host* Hasil *Scan* Menjadi Target *Sniffing*

2. Melakukan ARP *Poisoning* dengan *ettercap* terhadap jaringan untuk memberikan perintah mematikan fitur SSL pada iRaise.



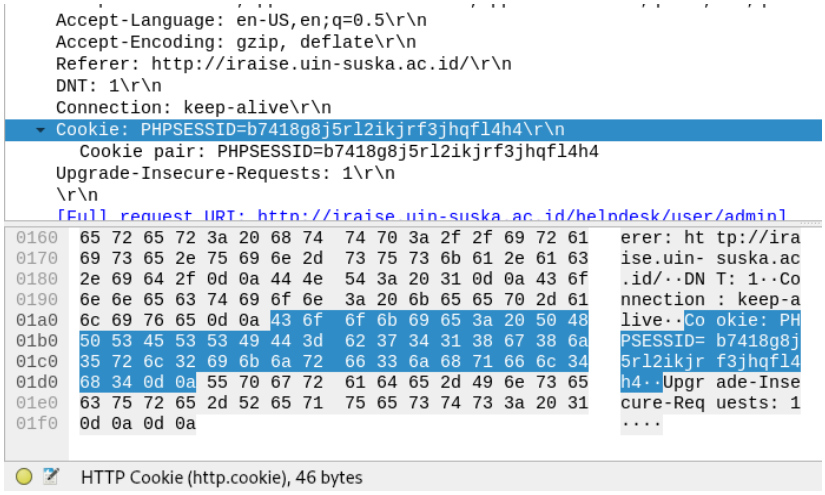
Gambar 4.14 ARP *Poisoning Ettercap*

3. Melakukan perintah mematikan SSL pada iRaise dengan terminal dengan perintah “sslstrip –a –f –k –l 10000”. Sebelum menjalankan perintah sslstrip dilakukan ip *forwarding* untuk mengalihkan akses ke port 10000, dikarenakan sslstrip berjalan pada port 10000 dengan perintah “iptables –t NAT –A PREROUTING –p –tcp –destination-port 80 –j REDIRECT –to-port 10000”.



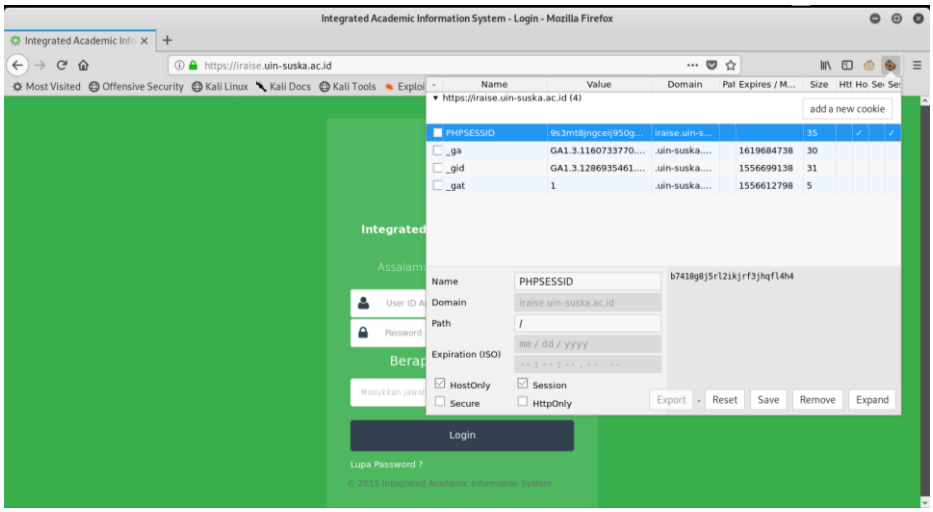
Gambar 4.15 Menjalankan SSLStrip

4. Setelah melakukan 3 tahapan sebelumnya tahapan terakhir adalah menunggu seorang admin untuk login dan memantau menggunakan *wireshark*. Terlihat pada gambar 4.16 seorang yang sedang melakukan *login* pada iRaise.



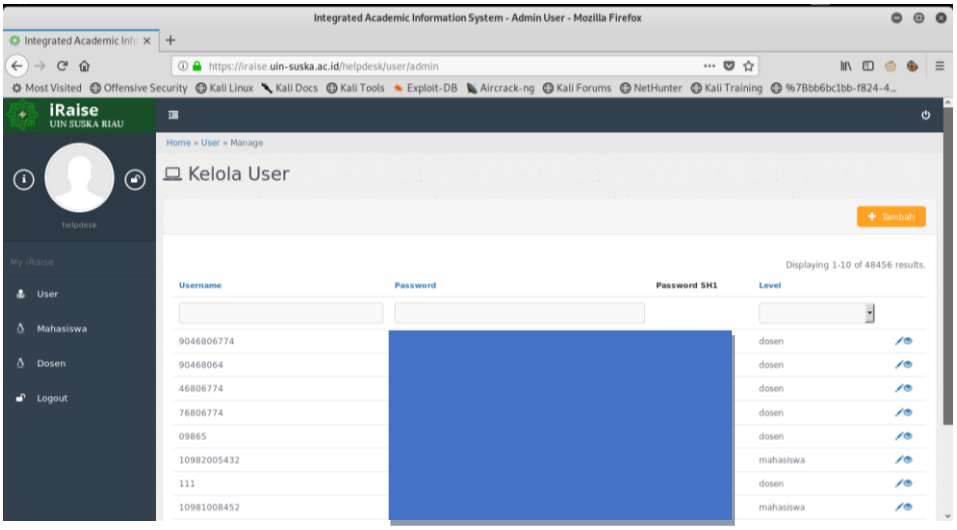
Gambar 4.16 Cookie iRaise

5. Dari hasil yang didapatkan menggunakan *wireshark*, penguji menggunkan bantuan plugin tambahan dari browser yaitu *Cookie Editor* seperti pada gambar 4.17



Gambar 4.17 Penggantian *Cookie* dengan *Cookie Editor*

6. Setelah mengganti *cookie* halaman *login* dengan *cookie* yang telah didapatkan menggunakan *wireshark* selanjutnya hanya dengan *reload* halaman iRaise dan sepeerti percobaan sebelumnya, penguji langsung diarahkan menuju halaman seorang yang diketahui adalah *helpdesk* seperti pada gambar 4.18



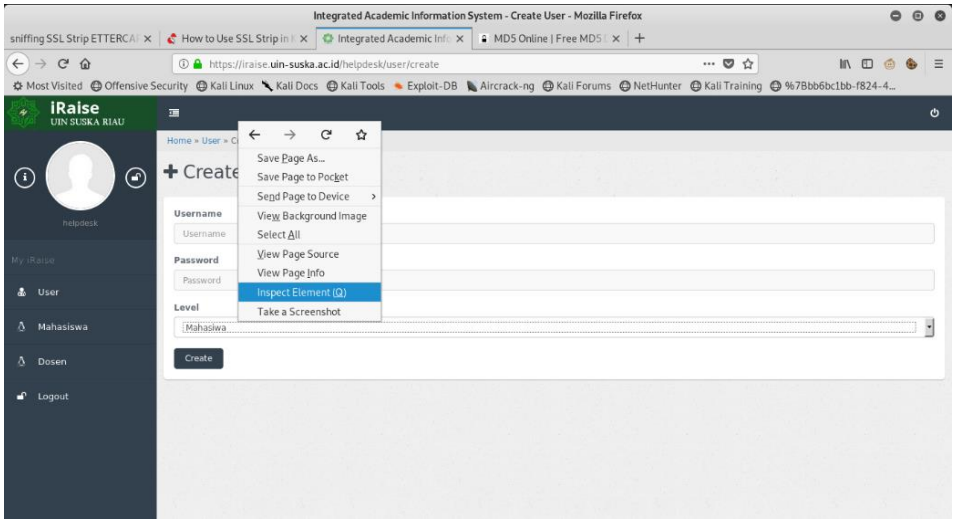
Gambar 4.18 Login Menggunakan *Cookie*

4.5.4. Percobaan Broken Access Control

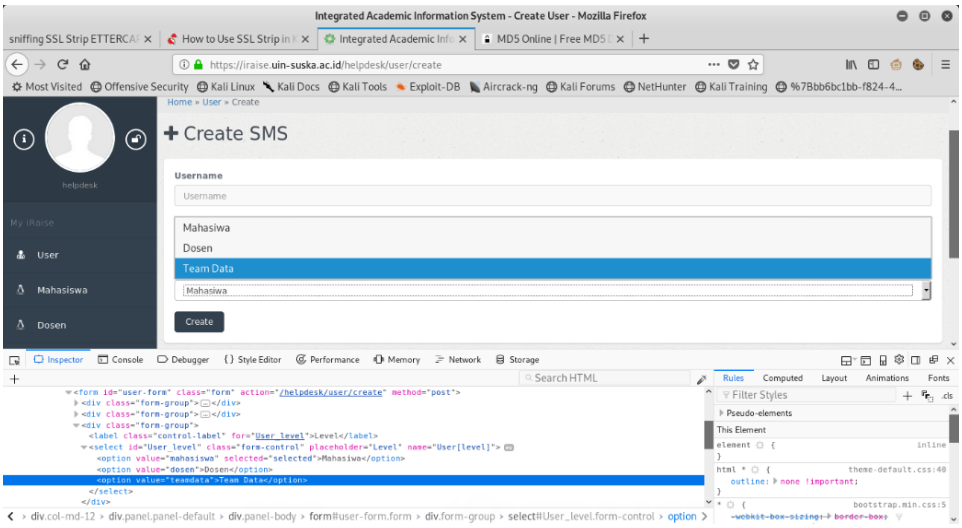
Setelah mendapatkan *cookie* seorang *user* yang ternyata adalah seorang *helpdesk*. Untuk menguji ancaman OWASP nomor 5 penguji melakukan percobaan mengakses halaman seorang *teamdata* dari akun dengan tingkatan *helpdesk* dengan mengganti URL dari kata *helpdesk* mengjadi *teamdata*. Setelah dilakukan percobaan dengan merubah URL dan selalu dialihkan kembali ke halaman *helpdesk*. Dan dapat dikatan iRaise tidak memilika ancaman pada OWASP nomor 5.

4.5.5. Percobaan Security Misconfiguration

Setelah mendapati akun *helpdesk* yang ternyata memiliki sebuah menu tambah *user*, akan dilakukan percobaan *security misconfiguration*. Setelah diakses menu tersebut hanya dapat menambahkan user dengan tingkatan mahasiswa dan dosen. Penguji mencoba melakukan merubah kode dengan bantuan *inspect element* dari *browser* yang digunakan. Penguji mencoba melakukan menambahkan pilihan *teamdata* setelah dosen dan mahasiswa seperti gambar 4.19 dan 4.20.

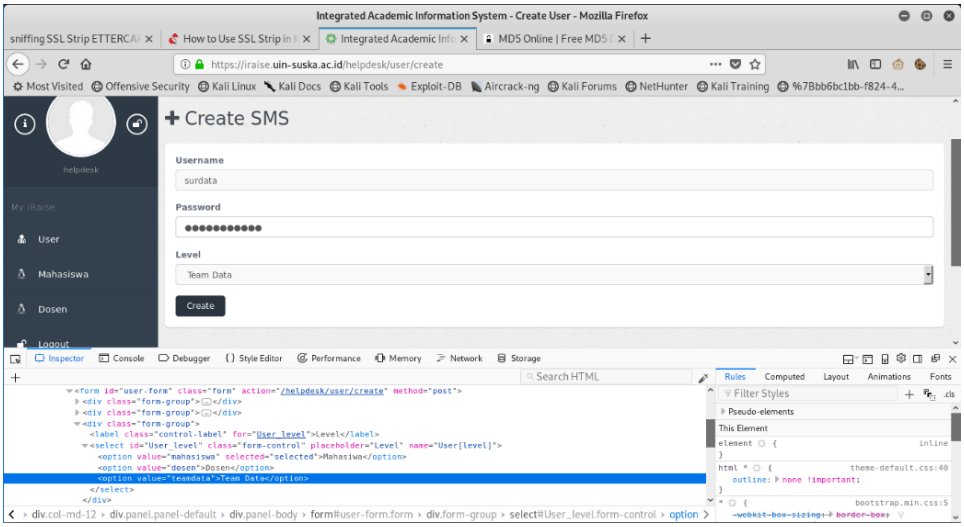


Gambar 4.19 *Inspect Element Menu Level*

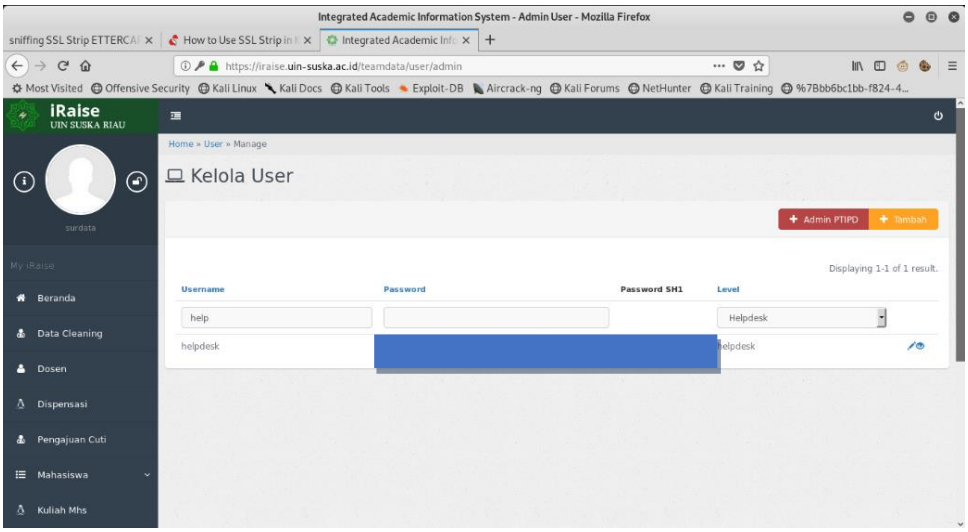


Gambar 4.20 Penambahan *Level Team Data* Dengan *Insepct Element*

Disini penguji melakukan percobaan menambahkan sebuah pilihan selain dosen dan mahasiswa, yaitu *teamdata*. Penguji menambahkan pilihan *teamdata* dikarenakan *teamdata* merupakan tingkatan yang sangat tinggi dan memiliki akses lebih banyak lagi dibandingkan dengan *helpdesk*. Setelah dilakukan penambahan, dan ternyata sukses melakukan penambahan *user* dengan tingkatan akses *teamdata*. Bukti sukses pembuatan *user* dengan tingkatan *teamdata* dapat dilihat pada gambar 4.21 dan 4.22



Gambar 4.21 Pembuatan *User* dengan tingkatan *Team Data*

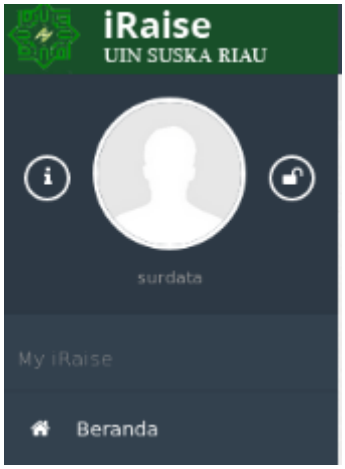


Gambar 4.22 User baru yang dibuat berhasil *login*

Hal ini juga sekaligus membuktikan bahwa iRaise memiliki ancaman OWASP nomor 6, yaitu *security misconfiguration* dikarenakan tidak melakukan konfigurasi yang bernar terhadap user *helpdesk* yang seharusnya tidak dapat menambahkan *user* dengan tingkatan yang lebih tinggi. Ini dapat menyebabkan *user* tersebut dapat mengakses yang tidak seharusnya bisa diakses olehnya. Dikarenakan penguji telah mendapati user dengan tingkatan akses yang sangat tinggi, penguji akan melanjutkan ke tahapan selanjutnya, yaitu *post exploitation*.

4.6. Post Exploitation

Post exploitation merupakan tindakan untuk mempertahankan akses yang telah didapatkan. Hal yang dilakukan pada tahapan ini adalah membuat user dengan nama palsu dan hampir menyerupai akun – akun admin. Ini merupakan tindakan yang paling aman dan tidak akan mudah untuk dicurigai oleh admin dari iRaise.Sebelumnya diketahui ada peretas iRaise yang membuat *user* dengan nama yang mencurigakan dengan tingkat akses yang sangat tinggi. Berdasarkan tindakan tersebut mudah diketahui dan berujung menghilangkan akses yang telah didapatkan, hal tersebut tidak dilakukan oleh penguji. Cukup dengan membuat user yang dapat membuat akun baru dan menyamarkan nama – nama user tersebut dengan nama seorang admin dari PTIPD seperti gambar 4.23.

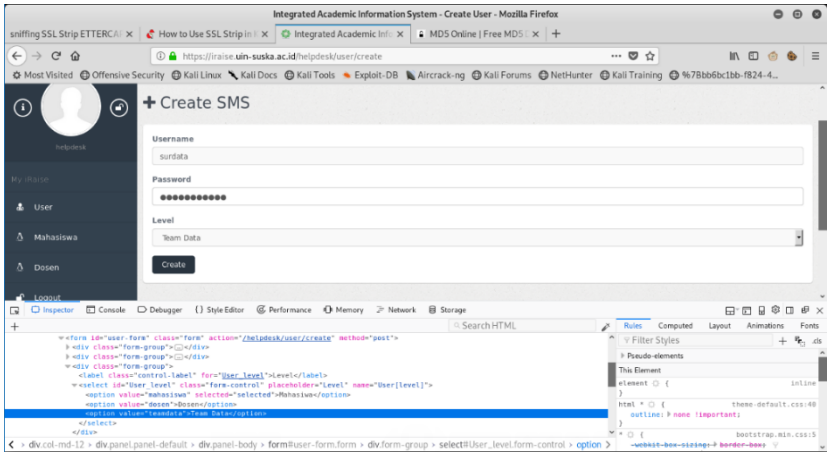


Gambar 4.23 User dengan nama mirip pegawai PTIPD

Jika hal tersebut dilakukan dan akan memberikan keuntungan dapat mempertahankan akses untuk jangka waktu yang lama dari user yang berhasil dibuat dengan tingkatan *teamdata* yang memiliki akses lebih tinggi, penguji mendapati *password helpdesk* yang sebelumnya digunakan untuk membuat akun dengan tingkatan *teamdata*. Disini tidak perlu merubah penguji tidak melakukan perubahan pada akun helpdesk, karena akun ini bisa dimanfaatkan untuk kembali membuat *user* dengan tingkatan *teamdata* jika akses *teamdata* hilang atau diketahui.

Untuk tahapan mendapatkan akses yang lebih tinggi lagi jika ada, cukup dengan memanfaatkan ancaman OWASP nomor 6 yaitu *security* misconfiguration.

Dengan melakukan inspect element pada browser dan menambahakan jenis tingkatan tersebut seperti gambar 4.24.



Gambar 4.24 Insepect Element

Dikarenakan teamdata adalah akses tertinggi jadi tidak perlu dilakukan mendapatkan akses yang lebih lagi.